Supporting Architecture Documentation: A Comparison of Two Ontologies for Knowledge Retrieval

de Graaf, Klaas Andries; Liang, Peng; Tang, Anthony; van Vliet, Hans

published in
Proceedings 19th International Conference on Evaluation and Assessment in Software Engineering (EASE2015) 2015

DOI (link to publisher)
10.1145/2745802.2745804

document version
Peer reviewed version

Supporting Architecture Documentation: A Comparison of Two Ontologies for Knowledge Retrieval

Klaas Andries de Graaf
VU University Amsterdam
ka.de.graaf@vu.nl

Peng Liang
Wuhan University
liangp@whu.edu.cn

Antony Tang
Swinburne University of Technology
atang@ict.swin.edu.au

Hans van Vliet
VU University Amsterdam
hans@cs.vu.nl

ABSTRACT

Context: Software architecture documentation is used to communicate architectural knowledge. It is often difficult for document users to find all the architectural knowledge they need to do their tasks, and this results in wasted time and mistakes during development. **Objective:** In this paper we investigate how ontology-based documentation may support users in finding the architectural knowledge they need. **Method:** We executed a controlled experiment to test for differences in knowledge retrieval efficiency and effectiveness between two groups of master students that used two ontologies built from different understandings of the architectural knowledge needs of document users. **Results:** Use of the ontology built based on a better understanding of architectural knowledge needs was significantly more efficient or effective for retrieving part of the knowledge needed by document users. We analysed participants' search actions and identified which organisation of knowledge in the ontologies resulted in efficient and effective knowledge retrieval. **Conclusion:** We found that an improved understanding of knowledge needs allows for the construction of an ontology from which document users retrieve knowledge more efficiently and effectively. In some cases we found that the ontology support for knowledge needs had to be traded off against ontology design criteria.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Search process; D.2.7 [**Software Engineering**]: Distribution, Maintenance, and Enhancement—*Documentation*

Keywords

Software architecture documentation, knowledge retrieval, ontology engineering, ontology-based documentation.

EASE 2015

1. INTRODUCTION

It is recognized by Bass *et al.* in [3] and Clements *et al.* in [6] that even a perfect Software Architecture (SA) is essentially useless if it is not understood; proper documentation should have enough detail, no ambiguity, and it must be organised such that users can quickly find information and answer their questions [19]. Documentation of software architecture serves three important purposes: it is used for education, system analysis, and it is the primary vehicle for stakeholder communication [6].

Architectural Knowledge (AK) is contained in SA documentation. AK can be defined as "*the integrated representation of the software architecture of a software-intensive system (or a family of systems), the architectural design decisions, and the external context/environment*" [15]. File-based documents, e.g., text and diagram files, are commonly used in industry practice to organise and retrieve AK [20].

File-based SA documentation is often 'one-size-fits-all' and does not serve the needs of specific users and their tasks well [20]. This mismatch between the expected and actual AK needs of document users reduces the value of SA documentation [17]. SA documentation that is not fitting for its users is not cost-effective [6].

The linear organisation of file-based documentation limits document writers in organising AK and explicitly communicating the relationships between AK that users need to find [9]. Relationships between AK are often tacit, e.g., between requirements and design rationale [22]. Industry practitioners consider incomplete documentation and missing relationships between AK, i.e., lack of 'traceability' between AK, as the main problems with SA documentation [20] and indicate that these problems negatively affect software maintainance [5]. Improving the traceability between AK leads to better architectural understanding [13, 21].

Ontology-based documentation makes use of ontologies for non-linear organisation of AK via classes and relationships, and this allows document writers to comprehensively organise AK and relationships between AK for the needs of document users. Recent studies provide evidence that the use of ontology-based AK organisation improves architectural review quality [12] as well as the efficiency and effectiveness of AK recovery [16] and AK retrieval [9], compared to file-based AK organisation.

However, we do not know how different ontologies perform in terms of their relative efficiency and effectiveness.

In a previous study [23], an ontology was built for the AK needs based on expected use cases for finding requirements and design. If we know the actual AK needs, and build an ontology according to these needs, how would this ontology perform? Would an ontology built from actual AK needs result in more efficient and effective AK retrieval than an ontology built from expected AK needs?

As such, we compare an ontology built from expected AK needs with an ontology built from actual AK needs. We investigate if there is any difference between the two ontologies in terms of AK retrieval efficiency and effectiveness. We also investigated the limitations of building an ontology from actual AK needs.

We report on a controlled experiment in which participants answer questions about AK from ontology-based SA documentation in a semantic wiki. The participants were organised in two groups. One group retrieved AK using an ontology built from the AK organisation in an SA document. Another group retrieved AK using an ontology built from the AK needs in an architectural review approach. Both ontologies were intended and used for architectural review [4], however, one was built from the *expected* AK needs of document users, i.e., built without knowing their exact AK needs and questions, and the other was built from the *actual* AK needs of document users in the experiment. We compared how the use of the two different ontologies affected the time-efficiency and effectiveness (in precision and recall) of AK retrieval by participants.

We identified which parts of the ontology-based AK organisation supported the experiment questions. By analysing the search actions of experiment participants we verified that the use of supporting AK organisation positively correlates with the efficiency and effectiveness of AK retrieval. We compared the AK organisation of the two ontologies and reflect on the use of several criteria for designing clear and extendible knowledge sharing ontologies, and how the use of these design criteria affected the support for AK needs in the constructed ontologies.

This paper makes the following contributions:

- Demonstrate how an improved understanding of AK needs can be used to provide ontology support for more efficient and effective AK retrieval.
- Identify the kind of ontology-based AK organisation that improves AK retrieval efficiency and effectiveness.
- Illustrate how the use of ontology design criteria affects the organisation and retrieval of AK.

In Section 2 we provide background on ontologies and ontology-based SA documentation. Section 3 details on the experiment and its results. In Section 4 we analyse the ontology-based AK organisation to explain the underlying causes for the experiment results. In Section 5 we discuss insights gained from constructing the ontologies and analysing the experiment results. Section 6 describes threats to validity and Section 7 reports our conclusions and future work.

2. BACKGROUND

2.1 Ontology-based SA Documentation

"An ontology" explicitly specifies the conceptualization of a domain [11], i.e., "an ontology" refers to a formal domain model in which concepts and relationships among concepts are described [16]. Ontologies enable a hierarchical classification of interrelated domain concepts and can be represented using a Resource Description Framework (RDF) Schema or the more expressive Web Ontology Language (OWL). The use of RDF makes ontologies human readable and machine-interpretable, allowing querying of and inference over knowledge.

The classes and relationships in an ontology can be used for organising AK. Each distinct ontology class and relationship has properties and descriptions that explicitly define their meaning, allowing different AK users to interpret them consistently and unambiguously. Relationships in an ontology allow its users to see how AK instances are interrelated, e.g., "*requirement X is realized by component Y*", and thereby improves traceability between AK. Moreover, the relationships between AK can be used to provide different views on, and cross-section of the same AK to satisfy the AK needs of diverse SA documentation users. The use of ontologies for SA documentation is described in more detail in [7–9, 12, 16, 17].

2.2 ArchiMind Semantic Wiki

A semantic wiki allows for presentation and navigation of classes and relationships in an ontology. Semantic wikis can provide features of traditional wiki-like systems, e.g., centralised access to and editing of documentation, versioning, and collaboration mechanisms.

We used the OntoWiki tool [2] as the basis for the tool in our ontology-based documentation approach. OntoWiki is open source and aims to support collaborative knowledge engineering. It offers web-based visualization and management of (ontology and its instances in) knowledge bases and semantic-enhanced search facilities. We made adaptations to version 0.9.5 of OntoWiki in order to optimize it for storage and retrieval of architecture documentation. We named the adapted version 'ArchiMind' (see `http://www.archimind.nl/archimind/` for a demo).

File-based documentation content, e.g., from word processors and UML tools, and its layout is stored in wikipages using a WYSIWYG editor in ArchiMind. 'Wikipage' is an ontology class and its instances are used to store documentation content. ArchiMind allows for semantic annotation of phrases in documentation content that refer to AK instances. For example, in phrase "*D11: managers can use analytics GUI*" one could annotate "*D11*" as (an instance of class) decision, "*manager*" as stakeholder, and "*GUI*" as component. The annotated text on the wikipages is highlighted yellow and, when one clicks it, a pop-up menu shows the full description of the AK instance, its relationships to other AK instances, and to other wikipages that also mention the AK instance.

When a fragment of text is annotated on a wikipage, a relationship is created from the wikipages to the AK instance(s) that the annotated text fragment refers to, and vice versa. AK instances become traceable to the various fragments of documentation content (wikipages) that describe it and vice versa. This helps users to locate (sources of) AK descriptions. Please see [7] for more details.

3. AK RETRIEVAL EXPERIMENT

SA documentation often does not support the AK needs of all document users in its AK organisation [20]. It is hard

for document writers to accurately predict all AK needs of document users, and this introduces a mismatch between the AK needs that are supported in an SA document and the actual AK needs of document users. We investigate whether support for the actual AK needs of document users in an ontology affects the efficiency and effectiveness of AK retrieval from ontology-based SA documentation.

We execute an experiment in which participants use two different ontologies to answer questions about AK as part of an architecture review. One ontology is built based on the AK needs that document writers expect document users to have (an 'expected-needs ontology') when conducting the architectural review. The other ontology is built based on the actual AK needs of document users (an 'actual-needs ontology') when they conduct the architectural review. The actual AK needs were identified by analysing questions in the architectural review approach, and modelled in the actual-needs ontology to provide an organisation for retrieving this AK. We test whether there is a significant difference in time-efficiency and effectiveness (in recall and precision) of AK retrieval between the two ontologies.

The primary experimental goals are:

- **(A)** evaluate the AK retrieval **efficiency** of an ontology built based on expected AK needs and an ontology built based on actual AK needs.
- **(B)** evaluate the AK retrieval **effectiveness** of an ontology built based on expected AK needs and an ontology built based on actual AK needs.

The experiment took place during a software architecture course given at the University of Amsterdam. This course is part of a professional software engineering master programme. The experiment was advocated to students during one of the final lectures as a voluntary extracurricular activity, and participation did not affect their course grades. In total 49 students participated in the experiment.

3.1 Experiment Materials

We used a file-based SA document for constructing the ontology-based documentation. This file-based SA document is the main deliverable of one team of 5 students in the software architecture course.

The SA document describes the architecture of a social network system for software developers that can answer queries of individual software developers and analyse the development community. Several students acted as stakeholders with specific concerns and requirements for the system. Students in the architect role had to design the system such that it addresses the concerns and requirements.

The SA document is written in English, consists of 39 pages, 16 (mostly UML) diagrams, 528 paragraphs, and 10.874 words. The AK in this document is organised by a table of contents with 26 (sub)sections. The document has a view-based AK organisation, containing a logical & implementation view and deployment view, following the architecture description principles of Bass *et al.* in [3] (advocated as textbook for the course), as well as [1, 14].

The content of the SA document was imported as wikipages in ArchiMind by following the process described in Section 2.2 and in [7]. AK elements and relationships between AK elements described in wikipage content were subsequently annotated using the two ontologies (expected-needs and actual-needs ontology) introduced in the next subsections.

3.1.1 Ontology Design Criteria

We followed design criteria for knowledge sharing ontologies, proposed by Gruber in [11], when constructing the two ontology used in the experiment. These criteria aim for construction of clear and extendible ontologies that are suitable for sharing knowledge in different applications.

Design criterion 1) on '*Clarity*' states that ontology constructs should be clear and objective, e.g., supported by natural language descriptions to specify their meaning.

Design criterion 2) on '*Coherence*' states that an ontology should be logically consistent, i.e., that its constructs and axioms do not contradict each other.

Design criterion 3) on '*Extendibility*' states that ontology extensions should be anticipated to support future tasks, and not require revision of existing ontology constructs.

Design criterion 4) on '*minimal encoding bias*' states that the definition of concepts should not depend on a particular encoding, implementation, or notation, because this restricts ontology use to specific programs and tools. Our use of ontology class '*Wikipage*', for storing SA document content in ArchiMind, is a form of encoding bias.

Design criterion 5) on '*minimal ontological commitment*' states that an ontology should model a domain using the minimal set of constructs and claims necessary for its knowledge sharing activities. This allows more freedom when extending and instantiating the ontology in different and specialized domains.

3.1.2 Expected-needs Ontology

We used the file-based SA document to construct an ontology for organising and indexing its AK in ontology-based documentation. We refer to this ontology as an '*expected-needs ontology*' because it was constructed based on the AK needs that the document writers *expected* document users to have. The expected-needs ontology is depicted in Figure 1 (without grey elements).

The 5 students who wrote the SA document knew that their architecture would be reviewed by the course staff for grading, and reviewed by other students as part of an assignment. The document writers expected architectural review by document users, and they understood that the AK needed for these review tasks should be recorded in their SA document. Similarly, the two researchers that built the ontology knew that the SA document was written with architectural review (by course staff and students) in mind. The researchers and the document authors however did not know the exact AK needs and actual questions that had to be answered during review.

The expected-needs ontology employed in the experiment was constructed based on the AK organisation of the file-based SA document, using the table of contents, views, tables, diagrams, lists of AK elements, and boldface headers, which make explicit where different types of AK and relationships between AK are recorded. Two researchers collaborated to identify AK concepts and relationships between AK based on the available AK organisation and their understanding of SA concepts.

For example, the table of contents in the SA document has subsections "*design rationale*", "*assumptions*", and "*scenarios*", which makes it explicit where design rationale is recorded. Boldface text and table column headers made design options explicit in the content of these sections. Based on this AK organisation we modelled class '*Design issue*',

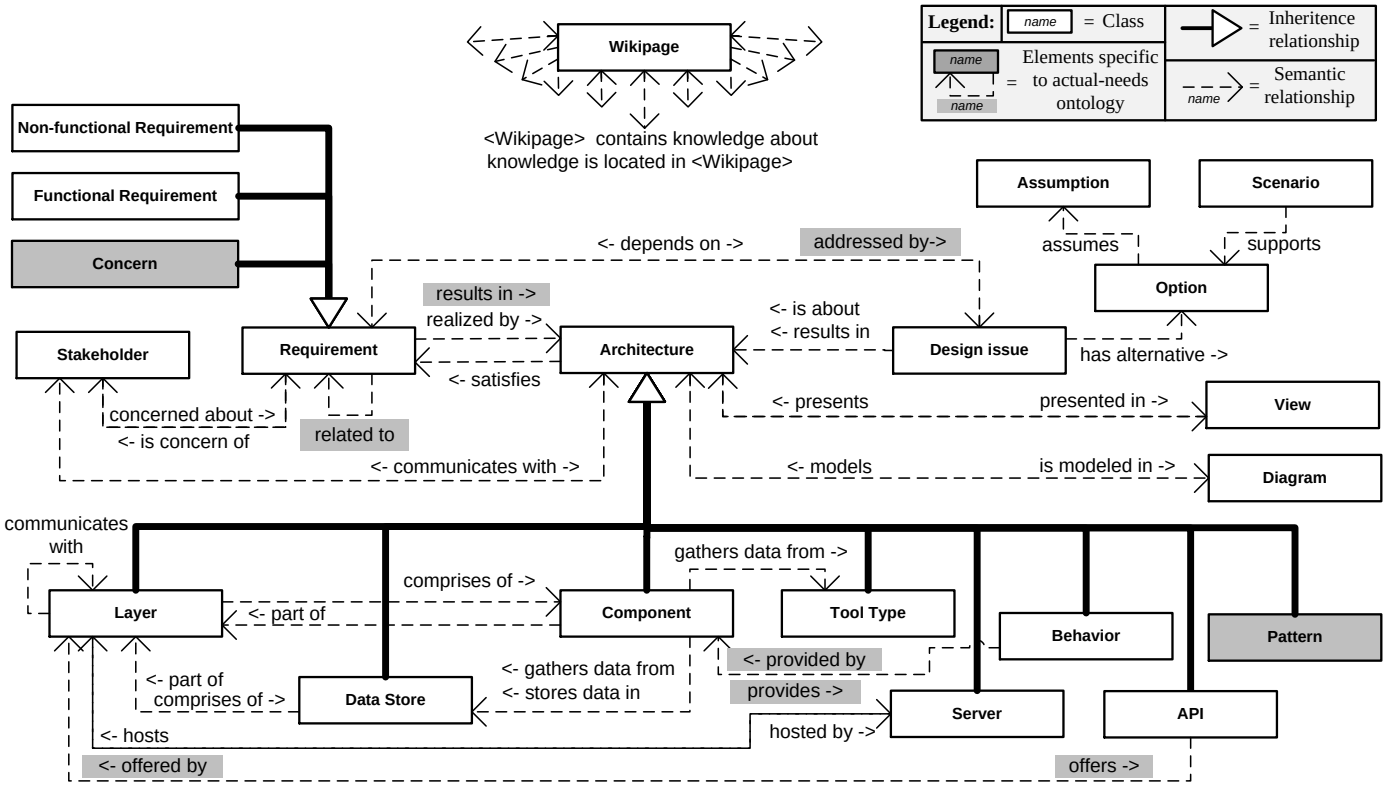

Figure 1: Expected-needs ontology (without grey elements) and actual-needs ontology (all elements)

'*Option*', '*Assumption*', and '*Scenario*', as well as relationships '*assumes*' and '*supports*' in the expected-needs ontology, as depicted in Figure 1. Various UML diagrams made elements of the architectural solution design explicit, e.g., components and layers, which were modelled as subclasses of '*Architecture*' in the ontology. In [12] a similar process was used to derive a domain ontology, during which AK elements were annotated in SA document content.

We followed the five design criteria, introduced in Section 3.1.1, whilst constructing the expected-needs ontology. Most notably, we modelled class '*Option*', '*Assumption*', and '*Scenario*', instead of a single class '*Decision*', when considering design criterion 3. This allows for an *extended* definition of design rationale and supports future associated tasks, e.g., extensive evaluation of design rationale, without having to revise the ontology.

3.1.3 Actual-needs Ontology

In [18] Nord *et al.* provide a number of questions, organised in question sets, to review architecture documentation as part of an SEI architectural review approach. 50 of the 127 SEI questions could be fully or partially answered from the SA document used in the experiment.

We used the AK needs expressed in these 50 SEI questions to build the actual-needs ontology. We identified AK types and relationships between AK in the SEI questions, and added these AK types and relationships to the actual-needs ontology if they were not yet present in the expected-needs ontology. This aims to support the AK needs of document users (experiment participants) when they apply the SEI architectural review approach.

For example, we added relationships '*provides*', '*offers*', and class '*Pattern*' to support document users in answering SEI questions about software development activities. We also added an ontology class '*Concern*' to support SEI questions for reviewing the documentation of stakeholders, concerns, and conformance to ISO/IEC 42010 [1].

The actual-needs ontology extends the expected-needs ontology. We chose for ontology extension, instead of building another ontology from scratch, because this allows us to investigate how the addition of specific ontology constructs to support AK needs influences AK retrieval. The added ontology constructs are marked grey in Figure 1.

The above process is in part similar to that in the 'typical question' approach to ontology engineering for software architecture documentation in [8]. In [8], questions that SA document users ask during their daily tasks, i.e., their 'typical questions' that represent their AK needs, are used to construct an ontology that supports the AK needs of document users.

We again used the ontology design criteria by Gruber [11] (see Section 3.1.1). For example, several of the SEI questions that review the support for software development activities required retrieval of implementation units and development dependencies. We however decided not to model implementation units and development dependencies as AK types and relationships in the ontology because this introduces a lot of *ontological commitment* (design criterion 5).

Several SEI questions for reviewing the documentation of stakeholders aim to locate stakeholder concerns in the SA document content, which is stored in wikipages in ArchiMind. We however decided to not add more relationships to and from class '*Wikipage*' because Wikipage is application specific and introduces *encoding bias* (design criterion 4).

We added relationship '*addressed by*', between class '*Requirement*' and '*Design issue*', as well as relationship '*results in*', between class '*Requirement*' and '*Architecture*' in the actual-needs ontology. This supports SEI questions for

reviewing the identification of requirements and design decisions. The added relationships are more specific than the existing relationships in the expected-needs ontology, and this adheres to design criterion 1) *clarity.*

3.1.4 Experiment Questions

We selected 5 SEI questions which are representative of the AK needs in the 50 SEI questions that were used to build the actual-needs ontology. These 5 SEI question are used as experiment questions to test and compare AK retrieval efficiency and effectiveness between the two ontologies. We used several criteria to select the 5 experiment questions from the 50 SEI questions.

Firstly, the experiment questions must be answerable from the SA documentation used in the experiment. Secondly, the experiment questions must have answers that can be quantitatively assessed, i.e., such that evaluators do not have to subjectively judge whether the answer to an open-ended question is either correct or incorrect. Thirdly, selected experiment questions should be representative of multiple SEI questions, to cover a large part of the SEI architectural review approach. We finally selected the following questions:

1: *List ten development dependencies between implementation units.*

2: *Which implementation units and decisions are explicitly related to requirement 'Security'?*

3: *Which architectural patterns are described in the architecture?*

4: *Which functional requirements are related to non-functional requirement 'Compatibility'?*

5: *Find ten wikipages in which the concerns of the stakeholders are addressed (not just listed).*

The experiment questions cover several aspects of the SEI architectural review approach. Question 1 reviews the support for software development in an SA document. Question 2 reviews the support for comprehensive architectural evaluation, software development, and identification of requirements and decisions in an SA document. Question 3 reviews the support for software development, and question 4 reviews the support for identification of requirements and design decisions. Question 5 reviews conformance to ISO/IEC 42010, support for comprehensive architectural evaluation, and reviews whether important stakeholders and concerns are captured.

We estimated that it was not acceptable for all participants to spend more than 45 minutes on the experiment. Therefore we limited the number of answers for experiment questions 1 and 5 to ten, and instantiated the requirements that have to be retrieved in questions 2 and 4. The original SEI questions require complete AK retrieval and thus more time. Question 3 is a shorter version of an SEI question.

3.2 Experiment Hypothesis

We formulate the following alternative hypotheses for experimental goals A and B specified in Section 3;

$\mathbf{H_{1A}}$ = *The use of the actual-needs ontology for answering experiment questions results in higher time-efficiency than the use of the expected-needs ontology.*

$\mathbf{H_{1B}}$ = *The use of the actual-needs ontology for answering experiment questions results in higher effectiveness than the use of the expected-needs ontology.*

The null hypotheses state that there is no difference in efficiency and effectiveness between the use of the two ontologies.

In the experiment we used one independent variable (or 'predictor variable') with two levels, namely the expected-needs and the actual-needs ontology. Two dependent variables (or 'response variables') are used in the experiment. **Time** is used as a measure of efficiency. The harmonic mean of precision and recall, the ***F*1 score**, introduced by van Rijsbergen in [24], is used for measuring effectiveness:

$$F1score = 2 * \frac{Precision \ * \ Recall}{Precision \ + \ Recall}$$

Where *Recall* is the proportion of relevant items retrieved from the total set of relevant items in a system and *precision* is the proportion of retrieved items that is relevant in a result set. *Recall* represents the completeness of AK retrieval (i.e, answers by participants to experiment questions) and *precision* represents the correctness of AK retrieval.

The students that wrote the SA document also participated in the experiment. Their answers to the experiment questions were only used to verify the relevancy of items, or 'ground truth', when calculating precision and recall.

3.3 Experiment Procedure

A ten-minute introduction to ontology-based SA documentation was given to participants during the final lecture of the software architecture course. A researcher explained the functions and GUI of ArchiMind using a graphical overview in presentation slides. The participants also received a printed form with a graphical overview of ArchiMind's GUI and an explanation of its functionality.

The participants received another printed form which depicted the ontology they had to use. This form also instructed participants: "*Please try to simulate your behavior in normal work and studies: balance between time-efficiency and correct answers.*". This instruction was given to encourage participants to perform the experiment seriously and answer the questions as time-effectively as possible.

We handed out two versions of the printed experiment instructions in random order. In one version of the instructions we asked participants to navigate to a URL which hosted ArchiMind with the expected-needs ontology. The other version led participants to a URL which hosted ArchiMind with the actual-needs ontology. These URLs also hosted a web-based form which listed the questions, input boxes for answers, and the expected format of answers. The experiment took place in two classrooms with PCs containing the same hardware and operating system image.

Before the start of the experiment we asked each participant: "*How many years of working experience in IT industry do you have?*". Participants reported an average of 3.92 years experience in IT industry. 27 participants used the expected-needs ontology and reported 4.78 years experience on average, and 22 participants used the actual-needs ontology and reported 2.87 years of experience on average. Two participants using the expected-needs ontology were outliers with 16 and 30 years of experience, which explains the large difference in average years of experience between the two groups.

Table 1: Time-Efficiency (Seconds), Effectiveness (F1 Score), and Statistical Test Results in Experiment

Experiment question	Metric (for each table row)	Average expected-needs	Average actual-needs	Difference	*p-value* test results	Number of measurements	Effect size *r*
1	Seconds	1094	1278	184	0.26102	40	0.18
	F1score	0.22	0.32	0.10	0.34867	43	0.14
2	Seconds	349	376	27	0.53151	36	0.10
	F1score	0.72	0.63	0.09	0.11478	38	0.26
3	Seconds	246	117	129	0.00082	30	0.61
	F1score	0.24	0.79	0.56	0.00010	34	0.67
4	Seconds	290	229	61	0.39308	27	0.16
	F1score	0.08	0.75	0.67	0.00003	33	0.73
5	Seconds	448	366	82	0.41236	24	0.17
	F1score	0.72	0.60	0.12	0.23968	29	0.22

3.4 Experiment Test Results

Using the Shapiro-Wilk and Kolmogorov-Smirnov tests, we found that the experiment measurements were not normally distributed. Therefore we applied the non-parametric Mann-Whitney-Wilcoxon (MWW) test to the experiment measurements. Table 1 reports the average efficiency and effectiveness measurements, the *p-values* for one-tailed MWW tests (corrected for ties), and effect size per question.

12 participants encountered errors in ArchiMind whilst answering questions (the database server could not handle 49 concurrent users). We excluded measurements for these questions. In some cases we could include the F1 scores of answers in our measurements because participants lost time due to an error, but could still continue to give an answer. This results in an unpaired number of measurements (see second-to-last column of Table 1) between the control and test group, however, the MWW test allows testing of unpaired measurements.

Column '*p-value test results*' in Table 1 contains the *p-values* for one-tailed MWW test results on efficiency for each row with '*Seconds*' in column '*Metric*'. Table 1 shows that the difference in AK retrieval efficiency between the expected-needs and actual-needs ontology is statistically insignificant at the p=0.05 level for all experiment questions, except for question 3. Consequently, we only reject the null hypothesis $\mathbf{H_{0A}}$ and accept the alternative hypothesis $\mathbf{H_{1A}}$ for question 3. Most participants that used the expected-needs ontology quickly gave up searching without finding answers to question 4. This explains the insignificant difference in efficiency between the ontologies for question 4.

Table 1 shows that the difference in AK retrieval effectiveness (rows with "*F1score*") between the expected-needs and actual-needs ontology is statistically significant at the p=0.05 level for experiment questions 3 and 4. Consequently, we reject the null hypothesis $\mathbf{H_{0B}}$ and accept the alternative hypothesis $\mathbf{H_{1B}}$ for questions 3 and 4.

4. AK ORGANISATION AND AK RETRIEVAL

In this section we analyse the experiment results. The actual-needs ontology provided significantly higher AK retrieval efficiency and effectiveness than the expected-needs ontology for experiment questions 3 and 4. The only intended (or 'designed') difference in the experiment materials was between the AK organisations provided by the two ontologies. We explain the differences in AK retrieval efficiency and effectiveness by comparing the difference in AK organisation that the two ontologies provided.

In Section 4.1 we analyse which ontology-based AK organisation supported participants in finding the AK and relationships between AK for each experiment question. We analyse the usage of supporting AK organisation from the search actions of participants in Section 4.2 and verify that this usage leads to efficient and effective AK retrieval. Section 4.3 describes how the AK retrieval efficiency and effectiveness of participants was affected by the supporting AK organisation for each experiment question and by different ontology constructs.

4.1 Fitting AK Organisation

The ontology-based documentation tested in the experiment is organised by ontology classes and relationships between classes. Figure 2 depicts the ontology-based AK organisation that provided one or more paths to the answers for each experiment question, i.e., the ontology classes with AK instances and the relationships between classes that can be used to navigate to AK instances and answers.

Some of the nodes on a path to the answer explicitly relate to the AK needs in the question asked. For example, experiment question 4 "*Which functional requirements are related to non-functional requirement 'Compatibility'?*", specifies that instances of AK types '*functional requirement*' and '*non-functional requirement*' have to be found. Both the actual-needs and expected-needs AK organisation in Figure 2 contain classes '*Functional Requirement*' and '*Non-functional Requirement*' which relate to the two AK types in question 4, and these classes can be used to find a path to answers for question 4.

Experiment question 4 also specifies a relationship '*related to*' between requirements, which is not supported in the expected-needs AK organisation. Participants can still identify the relationships between requirements in the content of wikipages, from properties of class *Requirement*, or by deriving the relationships via transitive or symmetric properties of other ontology relationships to and from requirements, e.g., '*depends on*', '*is concern of*', and '*realized by*'. The AK organisation provided by the actual-needs ontology additionally contains relationship '*related to*', which explicitly matches '*related to*' in experiment question 4 and provides a direct path to its answers.

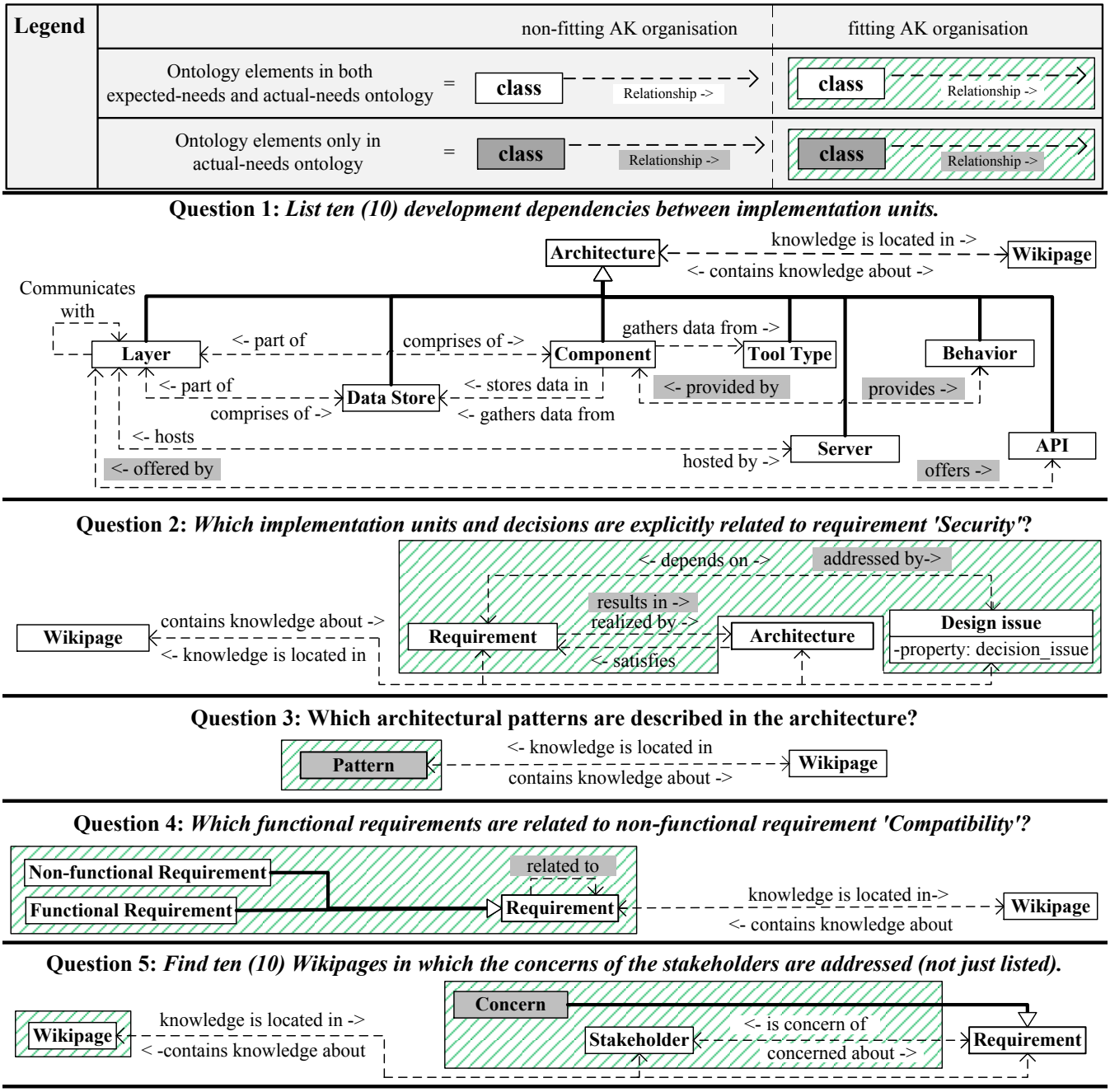

Figure 2: Ontology-based AK organisation that provided paths to answers for experiment questions

We term the elements in the ontology-based AK organisation that explicitly refer to the AK needs in the question asked, and provide a direct path to its answer(s), as "*fitting AK organisation*". We adopt the specific term "fitting AK organisation" and its definition because there may be other forms of support that an ontology-based AK organisation provides for AK needs in questions. Ontology classes and relationships that provided fitting AK organisation for AK needs in an experiment question are surrounded by boxes filled with green diagonal lines in Figure 2.

4.2 Usage of Fitting AK Organisation

There are various (confounding) factors that may have influenced the efficiency and effectiveness of AK retrieval in the experiment. For example, participants could find answers by keyword searching and reading plain text stored in wikipages, without making use of the fitting AK organisation provided by the ontologies. In this section we verify that the usage of fitting AK organisation influences the efficiency and effectiveness of AK retrieval.

We quantified the usage of AK organisation from over 3,000 AK retrieval actions of participants that were automatically logged during the experiment. Each logged search action is time-stamped and contains the name of the search action (e.g., '*keyword search*' and '*class navigation*'), the AK organisation that was used (e.g. class '*Requirement*'), and the names of visited AK instances or keyword search terms.

We measured the use of fitting AK organisation if 1) the

fitting AK organisation appeared on the screen of a participant for 3 seconds or more, and 2) the participant navigated to answers by following the fitting AK organisation or gave answers based on the fitting AK organisation shown on screen. We evaluated these criteria by looking at the properties of the logged search actions, their time-stamp, and by re-enacting the logged search actions in ArchiMind.

We measured usage of the fitting AK organisation based on the ontology classes and relationships, that provided fitting AK organisation for AK needs, depicted in Figure 2. This includes the use of fitting AK organisation in wikipage content, which was annotated based on the ontologies (see Section 2.2).

RatioTimeFitting is introduced as a metric to represent how much fitting AK organisation was used to answer an experiment question. *RatioTimeFitting* is calculated per participant per experiment question by dividing the '*time spent using fitting AK organisation*' by the '*total time spent searching for AK*'. *Time-effectiveness* is introduced in order to represent the efficiency and effectiveness of AK retrieval in a single metric. *Time-effectiveness* is calculated per answer in the experiment by dividing the *F1 score* (effectiveness) by the '*total time spent searching for AK*' (efficiency). A *Time-effectiveness* of 0.02 (e.g., for F1 score 1.0 divided by 50 seconds spent searching for AK) means that a participant was able to retrieve 2% of the complete and correct answer to an experiment question each second.

To verify that the use of fitting AK organisation influences the efficiency and effectiveness of AK retrieval, we test if a correlation exists between *RatioTimeFitting* and *Time-effectiveness*. We use the following hypothesis:

$\mathbf{H_{1C}}$ = *There is a correlation between RatioTimeFitting and Time-effectiveness.*
The null hypothesis states that there is no correlation.

Using the Kolmogorov-Smirnov test, we found that the measurements for *RatioTimeFitting* and *Time-effectiveness* are not normally distributed. Therefore the non-parametric Spearman's rank correlation test is applied.

Application of Spearman's rank test indicates a moderate positive correlation (coefficient $r = 0.509$) between *RatioTimeFitting* and *Time-effectiveness*. These test results are statistically significant at the p=0.01 level with a (2-sided) P-value of 1.005^{-11}. Consequently, we reject the null hypothesis $\mathbf{H_{0C}}$ and accept the alternative hypothesis $\mathbf{H_{1C}}$. This shows that the use of fitting AK organisation was positively correlated with the time-effectiveness of AK retrieval.

4.3 Fitting AK Organisation per Question

The experiment results in Table 1 show that there was no significant difference in AK retrieval efficiency and effectiveness between the two ontologies for questions 1 and 2. The insignificant difference for question 1 can be explained by the lack of fitting AK organisation in both ontologies, as shown in Figure 2. The actual-needs ontology contained four additional relationships that provided paths to answers for questions 1. The names of the four relationships did however not make it explicit to users where they could find development dependencies, and were thus not fitting for question 1.

The actual-needs ontology contained two additional relationships that provided fitting AK organisation for the AK needs in question 2. However, these two additional relationships in the actual-needs ontology were redundant because existing relationships '*depends on*', '*realized by*', and '*satisfies*' in both ontologies provided the same paths to answers and were fitting for the same AK needs in question 2. The actual-needs ontology thus provided redundant fitting AK organisation for AK needs in question 2, and its use did not result in significantly higher AK retrieval efficiency and effectiveness compared to use of the expected-needs ontology.

The actual-needs ontology provided additional fitting organisation by means of ontology class '*Pattern*' for AK needs in question 3 and relationship '*Related to*' (between requirements) for AK needs in question 4, as depicted in Figure 2. These AK needs were not yet supported by fitting AK organisation in the expected-needs ontology. Consequently, the AK retrieval efficiency and effectiveness of the actual-needs ontology was significantly higher than that of the expected-needs ontology for question 3, and AK retrieval effectiveness was significantly improved for question 4. This evidence also shows that a single class or relationship can be added to an ontology to provide fitting organisation for AK needs and thereby improve the time-effectiveness of AK retrieval.

The actual-needs ontology contains an additional class '*Concern*', which provided fitting AK organisation for AK needs in question 5. However, the expected-needs ontology already provided fitting AK organisation to support participants in finding the concerns for question 5, namely, via relationships '*is concern of*' and '*concerned about*'. Consequently, the AK retrieval efficiency and effectiveness of the actual-needs ontology was not significantly higher than that of the expected-needs ontology. Adding fitting AK organisation that is redundant to existing fitting AK organisation thus did not help to improve AK retrieval efficiency and effectiveness.

5. DISCUSSION

We built the expected-needs ontology based on the AK organisation of the SA document that was used in the experiment. The SA document contained all the AK needed by participants in the experiment because we only asked questions that could be answered from the SA document content. Moreover, the document authors anticipated the AK needs for architectural review, and architectural review questions were asked in the experiment. As such the SA document contained all AK needed in the experiment, which could have been modelled in the expected-needs ontology to provide fitting AK organisation.

However, certain types of AK and relationships between AK were not very explicit in the SA document, and difficult to identify without knowing if this AK was needed by document users. The AK in the SA document was organised to support the anticipated AK needs of its users, yet it did not clearly specify these AK needs. Building the expected-needs ontology thus required reverse engineering of the AK needs for which the SA document was written.

We noticed that the relationships between AK were especially hard to identify because they were not always explicitly described in the SA document. For example, the table of contents and boldface headers made it very explicit where decisions were recorded, yet their relationships with the requirements were less explicitly listed in the text inside the sections. This issue might be addressed by assuming that certain relationships between AK exist, instead of identifying and modeling the relationships between AK that are very explicit in the SA document.

We constructed the actual-needs ontology using the coding and ontology construction step in the 'typical question' approach described in [8]. It is suggested in [8] that the questions of users help AK ontology construction because they clearly identify the required types of AK and relationships between AK. This suggestion is supported by our study since we were able to identify AK types and relationships between AK from the SEI questions to construct the actual-needs ontology.

The actual-needs ontology however did not provide fitting AK organisation for all AK needs in the experiment, even though AK needs were identified from SEI questions. This lack of fitting AK organisation curtailed the efficiency and effectiveness of AK retrieval. The correlation found in Section 4.2 suggests that participants would have retrieved AK more efficiently and effectively if additional fitting AK organisation was provided to support their AK needs.

We found that the use of the ontology design criteria (see Section 3.1.1) restricted the modelling of the identified AK needs. The design criteria helped to construct a clear and extendible ontology for knowledge sharing across different applications. For example, based on these criteria we decided to comprehensively model design rationale to support future AK retrieval tasks (see Section 3.1.2). We also decided not to model certain constructs in the actual-needs ontology because they affected ontological commitment and encoding bias (see Section 3.1.3). This allows for more freedom to use the ontology in different domains and applications.

However, these decisions, made based on the ontology design criteria, restricted modelling of fitting AK organisation to support AK needs, and this in turn negatively affected the AK retrieval efficiency and effectiveness of participants. Use of the ontology design criteria was thus traded off against lower efficiency and effectiveness of AK retrieval.

Above findings can be used in practice to create ontology-based SA documentation from which AK can be retrieved efficiently and effectively. In [10] Falessi *et al.* investigated whether an accurate understanding of the value of AK for the activities of document users can be used to only document valuable AK, and thereby reduce the costs of producing documentation. Conversely, we investigated whether an accurate understanding of the AK needs of document users can be used to improve AK retrieval from documentation, and thereby reduce time-costs and increase the effectiveness of AK retrieval when consuming documentation.

6. THREATS TO VALIDITY

Internal Validity

The file-based SA documentation which was used as input for the experiment was written by 5 students, of which 4 participated in the experiment. We excluded the answers of these document authors from statistical testing because their prior knowledge about the SA documentation would bias the experiment results. Instead we used their answers as a 'gold standard' (i.e. 'ground truth') to verify that our evaluation of precision and recall was correct.

A group of 5 students who acted as stakeholders in the SA course answered several SEI questions from the file-based SA document as part of an assignment prior to the experiment. The AK needs in these questions partially overlaps with the AK needs in experiment questions, which means that these 5 students may have had prior knowledge about the answers to questions in the experiment. This threat to validity was mitigated by equal distribution of the 4 stakeholder-students that participated in the experiment across the test and control group: two students used the expected-needs ontology and two used the actual-needs ontology.

Two students registered as 'anonymous' for the experiment, and may have been part of the 5 stakeholders or 5 document authors discussed above. They however retrieved AK with average efficiency and effectiveness, which leads us to believe they do not have prior knowledge and are not part of the document authors or stakeholders discussed above.

External Validity

The specific set of questions from the SEI architectural review approach that were used in the experiment limits generalization of the experiment results and findings. The SA documentation used in the experiment was written by students that used guidelines in [1, 3, 14] for view-based architecture description. The experiment documentation can be generalized to documentation practice in industry to the extent that industry practitioners also make use of UML and view-based architecture descriptions.

The ArchiMind semantic wiki tool is not specific to the ontologies and SA documentation in the experiment, and can be used for any type of ontology and documentation. The findings in this work can to a certain extent be generalized to ontology-based documentation approaches that make use of a semantic wiki tool.

7. CONCLUSIONS

It is important that AK is quickly and correctly retrieved from SA documentation, however, SA documentation often does not support its users in retrieving all the AK that they need for their tasks. In this paper we investigated whether document users retrieve AK more efficiently and effectively when they use an ontology-based document organisation that is constructed from an improved understanding of their AK needs. We conducted an experiment to compare AK retrieval using an ontology built from the expected AK needs of document users and an ontology built from their actual AK needs.

The use of the actual-needs ontology was significantly more efficient and effective for answering one experiment question, and significantly more effective for answering another experiment question, compared to the use of the expected-needs ontology. Part of the ontology-based AK organisation was fitting for AK retrieval; it explicitly denoted the AK types and relationships between AK that participants needed to retrieve. We found that the use of AK organisation that was fitting for AK needs in a question had a significant correlation with the efficiency and effectiveness of answering that question. The actual-needs ontology provided more fitting AK organisation for the AK needs in two questions, and its users answered one of the two questions more efficiently and effectively and the other question more effectively than users of the expected-needs ontology.

The experiment results show that better support for the AK needs of ontology-based documentation users improves the efficiency and effectiveness when the users retrieve the AK that they need. The results highlight the importance of accurately understanding the AK needs of SA documentation users; it allows document writers to organise AK such that document users can efficiently and effectively retrieve

the AK that they need.

Not all AK needs were supported by fitting AK organisation, and this curtailed the efficiency and effectiveness of AK retrieval. This was caused by the use of criteria for designing clear and extendible knowledge sharing ontologies, which limited the amount of fitting AK organisation that was added to support AK needs. The use of the ontology design criteria was traded off against the efficiency and effectiveness of AK retrieval.

We plan to further investigate how fitting AK organisation can be provided for more AK needs of document users, whilst using design criteria for knowledge sharing ontologies.

Acknowledgements

The authors wish to thank the students that participated in the experiment during the 2012 Software Architecture course at the University of Amsterdam. This research has been partially sponsored by the Dutch "Regeling Kenniswerkers", project KWR09164, "Stephenson: Architecture knowledge sharing practices in software product lines for print systems" and by the Natural Science Foundation of China (NSFC) project No. 61170025 "KeSRAD: Knowledge-enabled Software Requirements to Architecture Documentation".

8. REFERENCES

[1] ISO/IEC/IEEE systems and software engineering – architecture description. *ISO/IEC/IEEE 42010:2011(E)*, 2011.

[2] S. Auer, S. Dietzold, and T. Riechert. Ontowiki a tool for social, semantic collaboration. In *International Semantic Web Conference (ISWC)*, volume 4273, pages 736–749. Springer LNCS, 2006.

[3] L. Bass, P. Clements, and R. Kazman. *Software Architecture in Practice*. Addison-Wesley Professional, 3rd edition, 2012.

[4] G. Booch. Architecture reviews. *IEEE Software*, 27(3):96, 95, 2010.

[5] J. C. Chen and S. J. Huang. An empirical analysis of the impact of software development problem factors on software maintainability. *Journal of Systems and Software*, 82(6):981–992, 2009.

[6] P. Clements, F. Bachmann, L. Bass, D. Garlan, J. Ivers, R. Little, R. Nord, and J. Stafford. *Documenting Software Architectures: Views and Beyond*. Addison-Wesley, 2002.

[7] K. A. de Graaf. Annotating software documentation in semantic wikis. In *Workshop on Exploiting semantic annotations in information retrieval (ESAIR)*, pages 5–6. ACM, 2011.

[8] K. A. de Graaf, P. Liang, A. Tang, W. R. van Hage, and H. van Vliet. An exploratory study on ontology engineering for software architecture documentation. *Computers in Industry*, 65(7):1053 – 1064, 2014.

[9] K. A. de Graaf, A. Tang, P. Liang, and H. van Vliet. Ontology-based software architecture documentation. In *Joint Working IEEE/IFIP Conference on Software Architecture (WICSA) and European Conference on Software Architecture (ECSA)*, pages 121–130. IEEE, 2012.

[10] D. Falessi, L. C. Briand, G. Cantone, R. Capilla, and P. Kruchten. The value of design rationale information. *ACM Transactions on Software Engineering and Methodology*, 22(3):21:1–21:32, 2013.

[11] T. R. Gruber. Toward principles for the design of ontologies used for knowledge sharing. *International Journal of Human-Computer Studies*, 43(5-6):907–928, 1995.

[12] A. Jansen, P. Avgeriou, and J. S. van der Ven. Enriching software architecture documentation. *Journal of Systems and Software*, 82(8):1232–1248, 2009.

[13] M. A. Javed and U. Zdun. The supportive effect of traceability links in architecture-level software understanding: Two controlled experiments. In *Working IEEE/IFIP Conference on Software Architecture (WICSA)*, pages 215–224. IEEE, 2014.

[14] H. Koning and H. van Vliet. Real-life IT architecture design reports and their relation to IEEE std 1471 stakeholders and concerns. *Automated Software Engineering*, 13(2):201–223, 2006.

[15] P. Lago and P. Avgeriou. First workshop on sharing and reusing architectural knowledge. *ACM SIGSOFT Software Engineering Notes*, 31(5):32–36, 2006.

[16] C. López, V. Codocedo, H. Astudillo, and L. M. Cysneiros. Bridging the gap between software architecture rationale formalisms and actual architecture documents: An ontology-driven approach. *Science of Computer Programming*, 77(1):66–80, 2012.

[17] M. Nicoletti, J. A. Díaz-Pace, S. Schiaffino, A. Tommasel, and D. Godoy. Personalized architectural documentation based on stakeholders' information needs. *Journal of Software Engineering Research and Development*, 2(9), 2014, doi:10.1186/s40411-014-0009-3.

[18] R. L. Nord, P. C. Clements, D. Emery, and R. Hilliard. A structured approach for reviewing architecture documentation. Technical Report CMU/SEI-2009-TN-030, SEI, Software Engineering Institute, Carnegie Mellon University, 2009.

[19] D. L. Parnas. Precise Documentation: The Key to Better Software. In *The Future of Software Engineering*, chapter 8, pages 125–148. Springer, 2011.

[20] D. Rost, M. Naab, C. Lima, and C. von Flach Garcia Chavez. Software architecture documentation for developers: A survey. In *European Conference on Software Architecture (ECSA)*, pages 72–88. Springer LNCS, 2013.

[21] M. Shahin, P. Liang, and Z. Li. Architectural design decision visualization for architecture design: preliminary results of a controlled experiment. In *Proceedings of the 5th European Conference on Software Architecture (ECSA): Companion Volume*, pages 2:1–2:8. ACM, 2011.

[22] A. Tang and M. F. Lau. Software architecture review by association. *Journal of Systems and Software*, 88(0):87–101, 2014.

[23] A. Tang, P. Liang, V. Clerc, and H. van Vliet. Traceability in the co-evolution of architectural requirements and design. In *Relating Software Requirements and Architectures*, chapter 4, pages 35 – 60. Springer, 2011.

[24] C. van Rijsbergen. *Information Retrieval*. Butterworths & Co, second edition, 1979.